Dynamic Trust Management for Delay Tolerant Networks and Its Application to Secure Routing

Ing-Ray Chen, *Member, IEEE* , Fenye Bao, MoonJeong Chang, and Jin-Hee Cho, *Member, IEEE*

Abstract—Delay tolerant networks (DTNs) are characterized by high end-to-end latency, frequent disconnection, and opportunistic communication over unreliable wireless links. In this paper, we design and validate a dynamic trust management protocol for secure routing optimization in DTN environments in the presence of well-behaved, selfish and malicious nodes. We develop a novel model-based methodology for the analysis of our trust protocol and validate it via extensive simulation. Moreover, we address dynamic trust management, i.e., determining and applying the best operational settings at runtime in response to dynamically changing network conditions to minimize trust bias and to maximize the routing application performance. We perform a comparative analysis of our proposed routing protocol against Bayesian trust-based and non-trust based (PROPHET and epidemic) routing protocols. The results demonstrate that our protocol is able to deal with selfish behaviors and is resilient against trust-related attacks. Furthermore, our trust-based routing protocol can effectively trade off message overhead and message delay for a significant gain in delivery ratio. Our trust-based routing protocol operating under identified best settings outperforms Bayesian trust-based routing and PROPHET, and approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message or protocol maintenance overhead.

Index Terms—Delay tolerant networks, dynamic trust management, secure routing, performance analysis, design and validation

1 Introduction

A delay tolerant network (DTN) comprises mobile nodes (e.g., humans in a social DTN) experiencing sparse connection, opportunistic communication, and frequently changing network topology. Because of lack of end-to-end connectivity, routing in DTN adopts a *store-carry-and-forward* scheme by which messages are forwarded through a number of intermediate nodes leveraging opportunistic encountering, hence resulting in high end-to-end latency.

In this paper, we propose *dynamic trust management* for DTNs to deal with both malicious and selfish misbehaving nodes. Our notion of selfishness is social selfishness [18], [21] as very often humans carrying communication devices (smart phones, GPSs, etc.) in a DTN are socially selfish to outsiders but unselfish to friends. Our notion of maliciousness refers to malicious nodes performing trust-related attacks to disrupt DTN operations built on trust (e.g., trust-based DTN routing considered in this paper). We aim to *design* and *validate* a dynamic trust management protocol for DTN routing performance optimization in response to dynamically changing conditions such as the population of misbehaving nodes.

- *I.-R. Chen, F. Bao, and M. Chang are with the Department of Computer Science, Virginia Polytechnic Institute and State University, 7054 Haycock Rd, Falls Church, VA 22043. E-mail: {irchen, baofenye, mjchang}@vt.edu.*
- *J.-H. Cho is with Computational and Information Sciences Directorate, US Army Research Laboratory, Powder Mill Rd. Adelphi, MD 20783. E-mail: jinhee.cho@us.army.mil.*

Manuscript received 20 Oct. 2012; revised 29 Jan. 2013; accepted 20 Mar. 2013; date of publication 11 Apr. 2013; date of current version 21 Mar. 2014. Recommended for acceptance D. Turgut.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.116

The contributions of the paper relative to existing work in trust/reputation management for DTNs are summarized as follows.

1. We propose to combine social trust deriving from social networks [25] and traditional quality of service (QoS) trust deriving from communication networks into a composite trust metric to assess the trust of a node in a DTN. To cope with both malicious and socially selfish nodes, we consider "healthiness" and "unselfishness" as two social trust metrics.
2. We propose the notions of "subjective trust" versus "objective trust" based on *ground truth* for protocol validation. For example, the healthiness trust of a good node should converge to 1 (ground truth) minus a false positive probability caused by noise, while the healthiness of a bad node should converge to 0 (ground truth) plus a false negative probability caused by noise and the random attack probability with which this bad node performs trust-related attacks.
3. We address the issue of application performance maximization (trust-based DTN routing in this paper) through *dynamic trust management* by adjusting trust aggregation/formation protocol settings dynamically in response to changing conditions to maximize DTN routing performance. Essentially we address the importance of integration of trust and security metrics into routing and replication decisions in DTNs.
4. We develop a novel model-based methodology utilizing stochastic petri net (SPN) techniques [26] for the analysis of our trust protocol and validate it via extensive simulation. The model validated with simulation yields actual *ground truth* node

status against which "subjective" trust obtained from executing the trust protocol is verified, and helps identify the best protocol settings in response to dynamically changing network conditions to minimize trust bias and to maximize the routing application performance.

5. We perform a comparative analysis of our trust-based DTN routing protocol built on top of dynamic trust management with simulation validation against routing based on Bayesian trust management [12], [14] (called Bayesian trust-based routing for short) and non-trust based (PROPHET [19] and epidemic [27]) protocols. Our trust-based routing protocol outperforms Bayesian trust-based routing and PROPHET. Further, it approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message or protocol maintenance overhead.

The rest of the paper is organized as follows. In Section 2, we survey existing trust management protocols and approaches to deal with misbehaving nodes in DTNs. In Section 3, we describe the system model. In Section 4, we describe our dynamic trust management protocol. In Section 5, we develop a performance model for the analysis of our trust protocol. In Section 6, we first identify the best protocol settings to minimize trust bias and to maximize the routing application performance, when given a set of parameters characterizing the operational and environmental conditions. Then we perform a comparative analysis of our proposed routing protocol against Bayesian trust-based routing and PROPHET. In Section 7, we validate our trust management protocol design through extensive simulation using both synthetic and real mobility data. In Section 8, we demonstrate the effectiveness of dynamic trust management in response to changing network conditions to maximize DTN routing performance. Finally in Section 9, we conclude the paper and discuss future research areas.

2 Related Work

We refer the readers to Appendix A of the supplemental file [6] for a comprehensive survey on the state of the art of trust management for DTNs.

3 System Model

We consider a DTN environment with no centralized trusted authority. Nodes communicate through multiple hops. When a node encounters another node, they exchange encounter histories certified by encounter tickets [16] so as to prevent black hole attacks to DTN routing. We differentiate socially selfish nodes from malicious nodes. A selfish node acts for its own interests including interests to its friends, groups, or communities. So it may drop packets arbitrarily just to save energy but it may decide to forward a packet if it has good social ties with the source, current carrier or destination node. We consider a friendship matrix [18] to represent the social ties among nodes. Each node keeps a friend list in its local storage. A similar concept to the friendship relationship is proposed in [20], where *familiar strangers* are identified based on colocation information in urban transport environments for media sharing. Our work is different from [20] in that rather than by frequent colocation instances, friendship is established by the existence of common friends. Energy spent for maintaining friend lists and executing matching operations is negligible because energy spent for computation is very small compared with that for DTN communication and matching operations are performed only when there is a change to the friend lists. When a node becomes selfish, it will only forward messages when it is a friend of the source, current carrier, or the destination node, while a well-behaved node performs altruistically regardless of the social ties. A malicious node aims to break the basic DTN routing functionality. In addition to dropping packets, a malicious node can perform the following trust-related attacks:

1. *Self-promoting attacks*: it can promote its importance (by providing good recommendations for itself) so as to attract packets routing through it (and being dropped).
2. *Bad-mouthing attacks*: it can ruin the reputation of well-behaved nodes (by providing bad recommendations against good nodes) so as to decrease the chance of packets routing through good nodes.
3. *Ballot stuffing*: it can boost the reputation of bad nodes (by providing good recommendations for them) so as to increase the chance of packets routing through malicious nodes (and being dropped).

A malicious attacker can perform random attacks to evade detection. We introduce a random attack probability P_{rand} to reflect random attack behavior. When $P_{rand} = 1$, the malicious attacker is a reckless attacker; when $P_{rand} < 1$ it is a random attacker.

A collaborative attack means that the malicious nodes in the system boost their allies and focus on particular victims in the system to victimize. Ballot stuffing and bad-mouthing attacks are a form of collaborative attacks to the trust system to boost the reputation of malicious nodes and to ruin the reputation of (and thus to victimize) good nodes. We mitigate collaborative attacks with an application-level trust optimization design by setting a trust recommender threshold T_{rec} to filter out less trustworthy recommenders, and a trust carrier threshold T_f to select trustworthy carriers for message forwarding. These two thresholds are dynamically changed in response to environment changes.

A node's trust value is assessed based on direct trust evaluation and indirect trust information like recommendations. The trust of one node toward another node is updated upon encounter events. Each node will execute the trust protocol independently and will perform its direct trust assessment toward an encountered node based on specific detection mechanisms designed for assessing a trust property X. Later in Section 4 we will discuss these specific detection mechanisms employed in our protocol for trust aggregation.

4 Trust Management Protocol

Our trust protocol considers trust composition, trust aggregation, trust formation and application-level trust optimization designs. Fig. 1 shows a flowchart of our

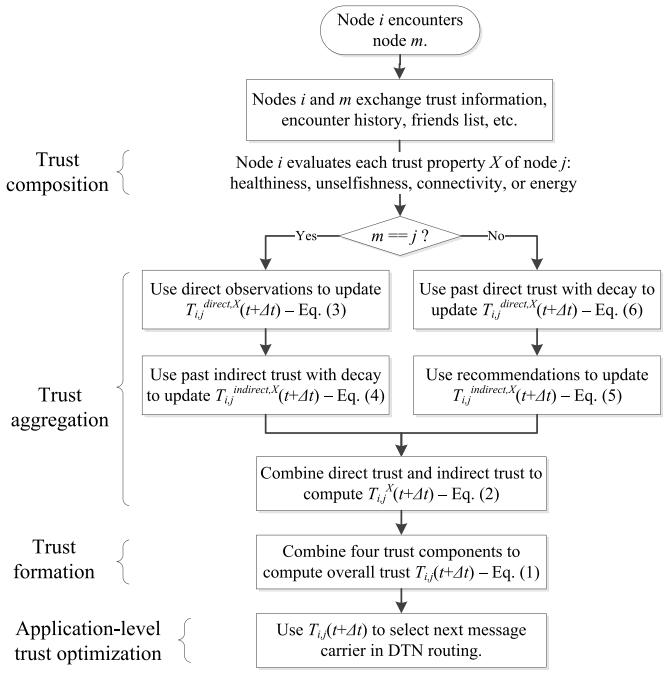


Fig. 1. A flowchart for trust protocol execution.

trust management protocol execution. For trust composition design (described in the top part of Fig. 1), we consider two types of trust properties:

- *QoS trust*: QoS trust [10] is evaluated through the communication network by the capability of a node to deliver messages to the destination node. We consider "connectivity" and "energy" to measure the QoS trust level of a node. The connectivity QoS trust is about the ability of a node to encounter other nodes due to its movement patterns. The energy QoS trust is about the battery energy of a node to perform the basic routing function.
- *Social trust*: Social trust [10], [25] is based on honesty or integrity in social relationships and friendship in social ties. We consider "healthiness" and social "unselfishness" to measure the social trust level of a node. The healthiness social trust is the belief of whether a node is malicious. The unselfishness social trust is the belief of whether a node is socially selfish. While social ties cover more than just friendship, we consider friendship as a major factor for determining a node's socially selfish behavior.

The selection of trust properties is application driven. In DTN routing, message delivery ratio and message delay are two important factors. We consider "healthiness", "unselfishness", and "energy" in order to achieve high message delivery ratio, and we consider "connectivity" to achieve low message delay.

We define a node's trust level as a real number in the range of [0, 1], with 1 indicating complete trust, 0.5 ignorance, and 0 complete distrust. We consider a trust formation design (described in the middle part of Fig. 1) by which the trust value of node j evaluated by node i at time t, denoted as $T_{i,j}(t)$, is computed by a weighted average of healthiness, unselfishness, connectivity, and energy as follows:

$$T_{i,j}(t) = \sum_{X}^{all} w^X \times T_{i,j}^X(t), \tag{1}$$

where X represents a trust property explored (X = healthiness, unselfishness, connectivity or energy), $T_{i,j}^X(t)$ is node i's trust in trust property X toward node j, and w^X is the weight associated with trust property X with the sum equal to 1. w^X is application-dependent. However, it is not related to the application priority [23] but dependent on the operational profile of an application [22].

In this paper, we aim to identify the best weight ratio under which the application performance (secure routing) is maximized, given an operational profile [22] as input. Before this can be achieved, however, one must address the accuracy issue of trust aggregation. That is, for each QoS or social trust property X, we must devise and validate the trust aggregation protocol executed by a trustor node to assess X of a trustee node such that the trust value computed is accurate with respect to actual status of the trustee node in X. This is achieved by devising a trust propagation protocol (described in the middle part of Fig. 1) with tunable parameters which can be adjusted based on each trust property.

When evaluating $T_{i,j}(t)$, we adopt the following notations: node i is the trustor, node j is the trustee, node m is a newly encountered node, and node k is a recommender. Node i (trustor) updates its trust toward node j (trustee) in trust property X upon encountering a node at time t over an encounter interval $[t, t+\Delta t]$ as follows:

$$T_{i,j}^X(t+\Delta t) = \beta T_{i,j}^{direct,X}(t+\Delta t) + (1-\beta) T_{i,j}^{indirect,X}(t+\Delta t). \tag{2}$$

In (2), $T_{i,j}^{direct,X}(t+\Delta t)$ and $T_{i,j}^{indirect,X}(t+\Delta t)$ are "direct trust" (based on direct observations) and "indirect trust" (based on recommendations) of node i toward node j in X at time $t+\Delta t$, respectively, and β in the range of [0, 1] is a parameter to weigh node i's own direct trust assessment toward node j. Every trust property X has its own specific β value under which subjective $T_{i,j}^X(t)$ obtained is accurate, i.e., close to actual status of node j in X at time t. Trust update is triggered by encounter events. Upon each encounter event, node i obtains either direct observations toward j (if node i encounters node j) or indirect recommendations towards node j (if node i encounters node $m, m \neq j$). This is indicated in the yes/no decision box in Fig. 1.

4.1 Trust Update upon Node i Encountering Node j

Upon encountering node j at time t, node i updates "direct trust" $T_{i,j}^{direct,X}(t+\Delta t)$ in (2) based on "direct" observations or interaction experiences with node j over the encounter interval $[t, t+\Delta t]$. When a monitoring node (node i) cannot properly monitor a trustee node (node j) upon encounter because of a short contact time, it adapts to this situation by discarding the current monitoring result and instead updating direct trust by its past direct trust toward j decayed

over the time interval Δt to model trust decay over time. Specifically, let $C_{i,j}^{direct,X}(t)$ be a boolean variable indicating if the needed data (discussed below) for assessing X is obtainable within Δt. Then, $T_{i,j}^{direct,X}(t+\Delta t)$, node i's trust in X toward node j at time $t+\Delta t$ upon encounter at time t, is calculated by:

$$T_{i,j}^{direct,X}(t+\Delta t) = \begin{cases} T_{i,j}^{encounter,X}(t+\Delta t), & if\ C_{i,j}^{direct,X}(t) = true \\ e^{-\lambda_d \Delta t} \times T_{i,j}^{direct,X}(t), & if\ C_{i,j}^{direct,X}(t) = false. \end{cases} \quad (3)$$

In other words, node i will update $T_{i,j}^{direct,X}(t+\Delta t)$ with its new direct trust toward node j in property X only if node i directly encounters node j at time t and the data needed for assessing X is obtainable within the encounter interval Δt; otherwise, node i will simply update $T_{i,j}^{direct,X}(t+\Delta t)$ with its past experience $T_{i,j}^{direct,X}(t)$ decayed overΔt. We adopt an exponential time decay factor, $e^{-\lambda_d \Delta t}$ (with $0 < \lambda_d \leq 0.1$ to limit the decay to at most 50 percent).

Node i assesses $T_{i,j}^{encounter,X}(t+\Delta t)$ based on data collected from direct observations toward node j over the encounter interval $[t, t+\Delta t]$ as follows:

- $T_{i,j}^{encounter,healthiness}(t+\Delta t)$: Node i assesses node j's unhealthiness based on evidences manifested due to malicious attacks including self-promoting, bad-mouthing and ballot stuffing attacks. Evidences of self-promoting attacks may be detected through the encounter history exchanged from node j. If the encounter history is not certified (e.g., using encounter tickets as in [4], [16]), or is certified but inconsistent with node i's encounter history matrix [11] accumulated, it is considered as a negative experience. A matrix element (j,k) records the number of times node j encountered node k, with each encounter being certified with an encounter ticket [4], [16] by both nodes j and k with time stamp information. Because of the encounter ticket mechanism, it is *impossible* that *node i*'s cumulative encounter history matrix element (j,k) is inconsistent with the encounter history provided by node j with node k if either node j or k is a good node, but it is *possible* that element (j,k) is inconsistent with the encounter history provided by node j with node k if both node j and node k are malicious, colluding and performing self-promoting attacks to attract packets to them. This is particularly the case when either node j or node k was good but later compromised and became malicious in between two encounters with node i. This inconsistency would be detected by *node i* and counted as one negative experience. Evidences of bad-mouthing/ballot stuffing attacks may be detected by comparing node j's recommendation toward another, say, node q, with the trust value of node i toward node q itself. If the percentage difference is higher than a threshold, it is considered suspicious and thus a negative experience. These positive/negative experiences are collected over the new encounter period $[t, t+\Delta t]$ to assess $T_{i,j}^{encounter,healthiness}(t+\Delta t)$. It is computed by the number of positive experiences over the total experiences in healthiness-related behavior.
- $T_{i,j}^{encounter,unselfishness}(t+\Delta t)$: Our notion of social selfishness is that friends will be cooperative toward each other even if they are selfish. Every node keeps a friend list and also adds itself as a member. When node i and node j encounter and directly interact with each other, if there is a change to either friend list, they can exchange their friend lists. To preserve privacy, node i and node j can agree on a one-way hash function (with a session key) upon encountering while exchanging the friend lists to hide the identities of their friends. This way, only common friends (the source node, node i, node j, or the destination node) will be identified while the identities of uncommon friends will not be revealed. From node i's perspective if node j is a friend of the source node, node i, or node d (the destination) then $T_{i,j}^{encounter,unselfishness}(t+\Delta t)$ is 1. Otherwise, node i will hope that node j is altruistic by examining the protocol compliance degree of node j. Specifically, node i applies monitoring techniques to detect altruistic behaviors, e.g., whether or not node j follows the prescribed protocol over $[t, t+\Delta t]$. Evidence of altruism is manifested by the behavior for executing beacon, encounter history exchange, packet receipt acknowledgement, and trust evaluation protocols expected out of node j. $T_{i,j}^{encounter,unselfishness}(t+\Delta t)$ is then computed by the number of positive experiences over the total experiences in unselfishness-related behavior. Here we note that node i will not monitor if node j has forwarded a packet since it is impractical to monitor packet forwarding in DTNs.
- $T_{i,j}^{encounter,connectivity}(t+\Delta t)$: While there is no predetermined connectivity pattern in DTNs, the connectivity of one node (j) to another node (d) is inherently associated with its mobility pattern and its social activities. This trust property represents the connectivity of node j to the destination node d. If the connectivity trust is high, then node j would be a good candidate for packet delivery to node d. Node i deduces node j's connectivity with node d based on its encounter matrix [11] collected over $[0, t+\Delta t]$, including the new encounter history received from node j. Specifically, node i uses its encounter history matrix accumulated over $[0, t+\Delta t]$ to compute $T_{i,j}^{encounter,connectivity}(t+\Delta t)$ as the ratio of the number of encounters between node j and node d to the maximum number of encounters between any node and node d. Note that node i should only accept a certified encounter history (as in [4], [16]) to avoid black hole attacks.
- $T_{i,j}^{encounter,energy}(t+\Delta t)$: This trust property represents the capability or competence of node j to do the basic routing function. Node i counts the ratio of the number of acknowledgement packets received from node j (at the MAC layer) over transmitted packets to node j, over $[t, t+\Delta t]$, to estimate energy status in node j.

In this case, since there is no new "indirect trust," node i simply updates $T_{i,j}^{indirect,X}(t+\Delta t)$ with its past experience $T_{i,j}^{indirect,X}(t)$ decayed over Δt, i.e.,

$$T_{i,j}^{indirect,X}(t+\Delta t) = e^{-\lambda_d \Delta t} \times T_{i,j}^{indirect,X}(t). \quad (4)$$

4.2 Trust Update upon Node i Encountering Node $m, m \neq j$

When node i encounters node $m, m \neq j$, node i uses its 1-hop neighbors (including node m) as recommenders to update "indirect trust" $T_{i,j}^{indirect,X}(t+\Delta t)$ in (2). An application-level optimization parameter is the recommender trust threshold T_{rec} for the selection of recommenders. Using T_{rec} provides robustness against bad-mouthing or ballot stuffing attacks since only recommendations from more trustworthy nodes are considered. The indirect trust evaluation toward node j is given in (5) below where R_i is the set containing node i's 1-hop neighbors with $T_{i,k}(t) \geq T_{rec}$ and $|R_i|$ indicates the cardinality of R_i. If node i considers node k as a trustworthy recommender, i.e., $T_{i,k}(t) \geq T_{rec}$, then node k is allowed to provide its recommendation to node i for evaluating node j. In this case, node i weighs node k's recommendation, $T_{k,j}^{X}(t)$, with node i's referral trust, $T_{i,k}^{X}(t)$, toward node k.

$$T_{i,j}^{indirect,X}(t+\Delta t) = \begin{cases} e^{-\lambda_d \Delta t} \times T_{i,j}^{indirect,X}(t), & if\ |R_i| = 0, \\ \frac{\sum_{k \in R_i} \{T_{i,k}^{X}(t) \times T_{k,j}^{X}(t)\}}{\sum_{k \in R_i} T_{i,k}^{X}(t)}, & if\ |R_i| > 0. \end{cases} \quad (5)$$

In this case, since there is no new "direct trust," node i simply updates $T_{i,j}^{direct,X}(t+\Delta t)$ with its past experience $T_{i,j}^{direct,X}(t)$ decayed over Δt, i.e.,

$$T_{i,j}^{direct,\ X}(t+\Delta t) = e^{-\lambda_d \Delta t} \times T_{i,j}^{direct,X}(t). \quad (6)$$

4.3 Application-Level Trust Optimization for Encounter-Based DTN Routing

When node i encounters node j, it uses $T_{i,j}(t)$ from Eq. (1) to decide whether or not node m can be the next message carrier to shorten message delay or improve message delivery ratio. We use two application-level optimization parameters for encounter-based DTN routing performance maximization. One parameter described earlier in Section 4.2 is the minimum trust threshold T_{rec} for the selection of recommenders. A high T_{rec} blocks bad-mouthing or ballot stuffing attacks but discourages recommendations, so "indirect trust" may be decayed unnecessarily because of lack of recommendations. A low T_{rec} on the other hand encourages recommendations but opens door to malicious attacks. Another application-level optimization parameter is the minimum trust threshold T_f for the selection of the next message carrier. Node i will forward the message to node j only if $T_{i,j}(t) \geq T_f$ and $T_{i,j}(t)$ is in the top Ω percentile among all $T_{i,m}(t)$'s. This helps the chance of selecting a trustworthy next message carrier. We aim to identify the best application-level trust optimization parameter settings in terms of T_{rec} and T_f to maximize the performance of the DTN routing application. This application-level trust optimization design is described in the bottom part of Fig. 1.

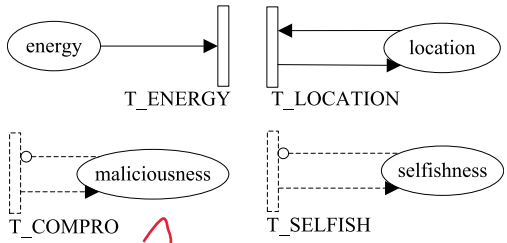


Fig. 2. SPN Model for a Node in the DTN.

5 Performance Modeling

We validate our trust management designs by a novel model-based analysis methodology via extensive simulation. Specifically we develop a mathematical model based on continuous-time semi-Markov stochastic processes (for which the event time may follow any general distribution) to define a DTN consisting of a large number of mobile nodes exhibiting heterogeneous social and QoS behaviors.

We take the concept of "operational profiles" in software reliability engineering [22] as we build the mathematical model. An operational profile is what the system expects to see during its operational phase. During the testing and debugging phase, a system would be tested with its anticipated operational profile to reveal design faults. Failures are detected and design faults causing system failures are removed to improve the system reliability. The operational profile of a DTN system specifies the operational and environmental conditions. Typically this would include knowledge regarding (a) hostility such as the expected percent of misbehaving nodes and if it is evolving the expected rate at which nodes become malicious or selfish or even the expected percent of misbehaving nodes as a function of time; (b) mobility traces providing information of how often nodes meet and interact with each other; (c) behavior specifications defining good behavior and misbehavior during protocol execution; and (d) resource information such as how fast energy is consumed.

We develop a probability model based on stochastic petri net techniques [26] to describe a DTN, given an operational profile as input. The SPN model for a DTN node is shown in Fig. 2 consisting of four places, namely, *energy*, *location*, *maliciousness* and *selfishness*. The underlying state machine is a semi-Markov model with 4-component states, i.e., (*energy*, *location*, *maliciousness*, *selfishness*), where *energy* is an integer holding the amount of energy left in the node, *location* is an integer holding the location of the node, *maliciousness* is a binary variable with 1 indicating the node is malicious and 0 otherwise, and *selfishness* is a binary variable with 1 indicating the node is socially selfish and 0 otherwise. A selfish node will forward a packet only if the source, current carrier or the destination is in its friend list. Here we note that a node's trust value actually is a real number in [0, 1]; it is calculated by a state-probability weighed sum of trust values assigned to the states of the

underlying semi-Markov model of the SPN performance model, i.e., trust value $= \sum_i$ (state probability of state $i \times$ trust value in state i). In some states, the trust value is binary. For example in a state in which a node is compromised, the trust value for property "healthiness" in this state is 0. Note that each node has its own SPN model. So there are as many SPN models as they are nodes in the DTN. The operational profile specifies the percent of malicious nodes and the percent of socially selfish nodes. Thus, some nodes will be malicious in accordance with this specification. Similarly some nodes will be selfish based on the percent of selfish nodes.

The purpose of the SPN model is to yield *ground truth* status of a node in terms of its healthiness, unselfishness, connectivity, and energy status. Then we can check *subjective trust* against *ground truth* status for validation of trust protocol designs. Below we explain how we leverage the SPN model to determine a node's ground truth status.

Location (Connectivity): The connectivity trust of node m toward node d is measured by the probability that both node m and node d are in the same location at time t. We use the *location subnet* to describe the location status of a node. Transition T_LOCATION is triggered when the node moves to a new area from its current location according to its mobility pattern. We consider both synthetic mobility models based on SWIM [15] and real mobility traces. This information along with the location information of other nodes at time t provides us the probability of two nodes encountering with each other at any time t.

Energy: We use the *energy subnet* to describe the energy status of a node. Place *energy* represents the current energy level of a node. An initial energy level (E_0) of each node represented by a number of tokens is assigned according to node heterogeneity information. A token is taken out when transition T_ENERGY fires representing the energy consumed during protocol execution, packet forwarding and/or performing attacks in the case of a malicious node. The rate of transition T_ENERGY indicates the energy consumption rate which varies depending on the ground truth status of the node (i.e., malicious or selfish). The operational profile specifies the energy consumption rate of a malicious node versus a selfish node versus a well-behaved node.

Healthiness: A malicious node is necessarily unhealthy. So we will know the ground truth status of healthiness of the node by simply inspecting if place *maliciousness* contains a token.

Unselfishness: A socially selfish node drops packets unless the source, current carrier or the destination node is in its friend list. We will know the ground truth status of unselfishness of the node by simply inspecting if place *selfishness* contains a token.

Dynamically changing environment conditions: With the goal to deal with malicious and selfish nodes in DTN routing, in this paper we consider a dynamically changing environment in which the number of misbehaving nodes (malicious or selfish) is changing over time. A node becomes malicious when it is captured and turned into a compromised node, as dictated by the per-node capture rate. The SPN output provides the probability that a node is compromised at time t. We model the capture event by a transition T_COMPRO (in dashed line) in Fig. 2. Once the transition T_COMPRO is triggered, a token will be moved into the place *maliciousness* representing that this node is compromised. Similarly, once the transition T_SELFISH (also in dashed line) is triggered, a token will be moved into the place *selfishness* representing that this node becomes selfish. The transition rates of T_COMPRO and T_SELFISH are λ_c and λ_s, respectively. We will use the SPN model augmented with the two dashed line transitions in Section 8 in which we treat the subject of dynamic trust management.

Objective trust evaluation: The SPN model described above yields actual or ground truth status of each node. The "objective" trust of node j at time t, denoted by $T_j(t)$, is also obtained from Eq. (1) except that $T_j^X(t)$ is being used instead of $T_{i,j}^X(t)$. Here $T_j^X(t)$ is simply the actual or ground truth status of node j in trust property X at time t obtainable from the SPN model for node j. The notion of "objective" trust evaluation is to validate subjective trust evaluation, that is, subjective trust evaluation is valid if the subjective trust value obtained as a result of executing our dynamic trust management protocol is accurate with respect to the objective trust value obtained from ground truth.

6 Numerical Results

In this section we present numerical results generated from the SPN model. Our trust evaluation results have two parts. The first part is about the convergence and accuracy of trust aggregation for individual trust properties. The second part is about maximizing application performance through trust formation (by setting the best weights to trust properties) and application-level trust optimization (by setting the best recommender trust threshold T_{rec}, and message carrier trust threshold T_f). Because different trust properties have their own intrinsic trust nature and react differently to trust decay over time, each trust property X has its own best set of (β, λ_d) under which $T_{i,j}^X(t)$ obtained from (2) would be the most accurate, i.e., closest to actual status of node j in trust property X, or $T_j^X(t)$. Recall that a higher β value indicates that subjective trust evaluation relies more on direct observations compared with indirect recommendations provided by the recommenders and that a higher λ_d indicates a higher trust decay rate. Once we ensure the accuracy of each trust property X, we can then address the trust formation issue, i.e., identifying the best way to form the overall trust out of QoS and social trust properties and the best way to set application-level trust parameters such that the application performance (i.e., secure routing) is maximized.

Table 1 lists a set of parameters and their values (for input parameters) as prescribed by the operational profile of a DTN. We consider $N = 20$ nodes moving according to the SWIM mobility model [15] modeling human social behaviors in an $m \times m = 16 \times 16$ $(4\text{ km} \times 4\text{ km})$ operational region, with each region coving $R = 250$ m radio radius. We use SWIM in this section for numerical results. Later in Section 7 we also use traces in our simulation studies. The initial energy of each node E_0 is set to 100 hours lifetime. The error probability of *direct trust* assessment of $T_{i,j}^{direct,X}(t + \Delta t)$ due

Table 1
System Parameters

Name	Value	Name	Value	Name	Value
$m \times m$	16×16 (4km×4km)	N	20	P_{error}	5%
Slope of SWIM	1.45	R	250m	P_{rand}	[0,1]
Pause of SWIM	≤ 4 hrs	E_0	100 hrs		

to environment noise denoted by P_{error} is set to 5 percent. For $X =$ healthiness, the false positive probability P_{fp} of misidentifying a healthy node as an unhealthy node is equivalent to P_{error}, i.e., 5 percent. For a compromised node performing random attacks with probability P_{rand} (in the range of [0, 1]) to evade detection, the false negative probability P_{fn} for missing an unhealthy node as a healthy node is $P_{error}P_{rand} + (1 - P_{error})(1 - P_{rand})$. That is, P_{fn} is P_{error} with probability P_{rand} (if attacking) and $1 - P_{error}$ with probability $1 - P_{rand}$ (if not attacking). We set $C_{i,j}^{direct,X}$ (in Eq. (3)) to true if the encounter duration is longer than 10 minutes as it would allow sufficient data to be collected for direct trust assessment of X; we set it to false otherwise. In SWIM [15], a node has a home location and a number of popular places. A node makes a move to one of the population places based on a prescribed pattern. The probability of a location being selected is higher if it is closer to the node's home location or if it has a higher *popularity* (visited by more nodes). When reaching the destination, the node pauses at the destination location for a period of time following a bounded power law distribution. We set the slope of the SWIM mobility model to 1.45 (as in [15]) and the upper-bound pause time to 4 hours.

The weight of direct trust (β), trust decay parameter (λ_d), trust threshold for the recommender (T_{rec}), trust threshold for the next carrier (T_f) and weight of trust property X (w^X) are design parameters whose best settings are to be determined as output. Here we should note that a social friendship matrix [18] and the percentages of selfish and malicious nodes, although not specified in Table 1, are also given as input, which we will vary in the analysis to test their effects on design parameters. Lastly, the node compromise rate (λ_c) and node selfishness rate (λ_s) for characterizing changing DTN conditions are also not specified in Table 1. We will consider these two parameters and treat the subject of dynamic trust management in Section 8.

6.1 Best Trust Propagation Protocol Settings to Minimize Trust Bias

Here we determine the best (β, λ_d) values that yield subjective trust evaluation closest to objective trust evaluation to minimize trust bias, given a set of parameter values as listed in Table 1 characterizing the operational and environmental conditions. We fix the percentage of selfish nodes to 30 percent and vary the percentage of malicious nodes from 0 to 45 percent to examine its effect. We set the recommender trust threshold (T_{rec}) to 0.6, since the trust value of a malicious node is likely to be lower than ignorance (0.5), so $T_{rec} \geq 0.6$ can effectively filter out false recommendations from malicious nodes. Since there are only two input parameters, we search the best (β, λ_d) for each trust property through exhaustive search, i.e., we compare subjective trust obtained through protocol execution under a given (β, λ_d) with objective trust. The best (β, λ_d) combination is the one that produces the lowest mean square error (MSE). This information determined at static time is recorded in a table to be used by dynamic trust management which we will discuss later in Section 8.

Table 2
Best (β, λ_d) to Minimize Trust Bias

% of malicious nodes	Healthiness (β, $\lambda_d \times 10^4$)	Unselfishness (β, $\lambda_d \times 10^4$)	Connectivity (β, $\lambda_d \times 10^4$)	Energy (β, $\lambda_d \times 10^4$)
0%	(0.44, 0.0)	(0.41, 0.2)	(0.80, 10)	(0.39, 0.1)
5%	(0.39, 0.0)	(0.41, 0.2)	(0.80, 10)	(0.39, 0.1)
10%	(0.40, 0.0)	(0.39, 0.0)	(0.80, 10)	(0.39, 0.1)
15%	(0.39, 0.0)	(0.37, 0.0)	(0.86, 10)	(0.39, 0.1)
20%	(0.41, 0.0)	(0.33, 0.0)	(0.91, 10)	(0.39, 0.1)
25%	(0.35, 0.0)	(0.30, 0.0)	(0.91, 10)	(0.48, 0.5)
30%	(0.35, 0.0)	(0.28, 0.0)	(0.91, 10)	(0.47, 0.0)
35%	(0.35, 0.0)	(0.26, 0.0)	(0.95, 10)	(0.49, 0.5)
40%	(0.35, 0.0)	(0.21, 0.1)	(0.95, 10)	(0.49, 0.5)
45%	(0.35, 0.0)	(0.22, 0.5)	(0.95, 10)	(0.58, 0.5)

In Table 2, we summarize the best (β, λ_d) values for each trust property for minimizing trust bias, given the percent of malicious nodes as input, for a trustor node (i.e., node i) randomly picked toward a trustee node (i.e., node j) also randomly picked. Each (β, λ_d) entry represents the best combination under which subjective trust $T_{i,j}^X(t)$ obtained as a result of executing our trust aggregation protocol for trust property X (as prescribed by Eq. (2)) deviates the least from objective trust for property X (that is, $T_j^X(t)$). We have observed for all cases the most deviation is 3 percent MSE. This substantiates our claim that there exists a distinct best protocol setting in terms of (β, λ_d) for each trust property X, with $X =$ connectivity, energy, healthiness or unselfishness. Furthermore, the best (β, λ_d) setting changes as the percent of misbehaving nodes changes dynamically.

6.2 Best Trust Formation Protocol Settings to Maximize Application Performance

Next we turn our attention to the trust formation issue to optimize application performance. For the secure routing application, two most important performance metrics are message delivery ratio and delay. In many situations, however, excessive long delays are not acceptable to DTN applications. We define the delivery ratio as the percentage of messages that are delivered successfully within an application deadline which is the maximum delay the application can tolerate. While our protocol is generic to any deadline, we set the deadline (or a time-to-live limit) to 2 hours to reveal the tradeoff between delay and delivery ratio in this environment setting for DTN routing. Our goal is to find the best way to assign the weight w^X to $X =$ healthiness, unselfishness, connectivity or energy to maximize the delivery ratio. Since the search space is small, we perform exhaust search to identify the best trust formation (w^X with an increment

Table 3
Best Trust Formation to Maximize Delivery Ratio

% of malicious nodes	$w^{healthiness}$	$w^{unselfishness}$	$w^{connectivity}$	w^{energy}
0%	0.0	0.6	0.4	0.0
5%	0.1	0.8	0.1	0.0
10%	0.3	0.3	0.3	0.1
15%	0.3	0.3	0.3	0.1
20%	0.3	0.3	0.3	0.1
25%	0.3	0.3	0.2	0.2
30%	0.3	0.3	0.3	0.1
35%	0.3	0.3	0.3	0.1
40%	0.4	0.3	0.2	0.1
45%	0.4	0.3	0.2	0.1

of 0.1) under which delivery ratio is maximized. This information again is recorded in a table to be used for dynamic trust management (Section 8). We assume that a malicious node drops all packets. A selfish node drops part of packets it receives depending on if it knows the source, current carrier or destination node socially (whether these nodes are in its friend list).

We consider two variations of secure routing protocols: single-copy forwarding ($L = 1$) and multi-copy forwarding ($L \geq 2$), where L is the maximum number of carriers to which a node can forward a message. Below we discuss how we identify the best setting for double-copy forwarding ($L = 2$). The best setting for other cases ($L = 1$ or $L > 2$) can be obtained in a similar way, but is not presented here due to space limitation.

Table 3 summarizes the best trust formation for maximizing delivery ratio under double-copy forwarding, given the percentage of malicious nodes as input. We first observe there is a distinct set of optimal weight settings under which delivery ratio is maximized. Second, the optimal weight of the *healthiness* trust property increases as the percent of malicious node increases. This is because in hostile environments, using a higher weight on *healthiness* helps identify malicious nodes to avoid message loss.

Fig. 3 correspondingly shows the maximum delivery ratio obtainable when the system operates under the best trust formation setting identified. We see that the delivery ratio remains high even as the percent of malicious nodes increases to as high as 45 percent. This to some extent demonstrates the resiliency property of our trust-based routing protocol against malicious attacks.

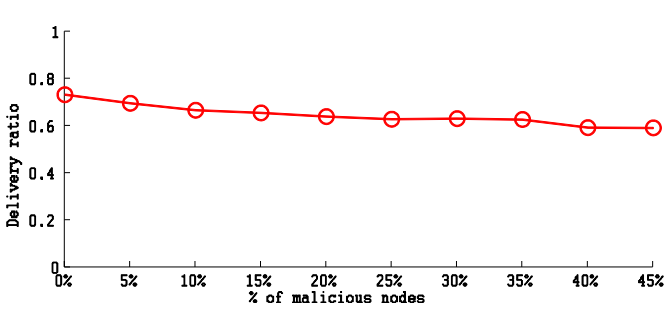


Fig. 3. Delivery ratio under best trust formation.

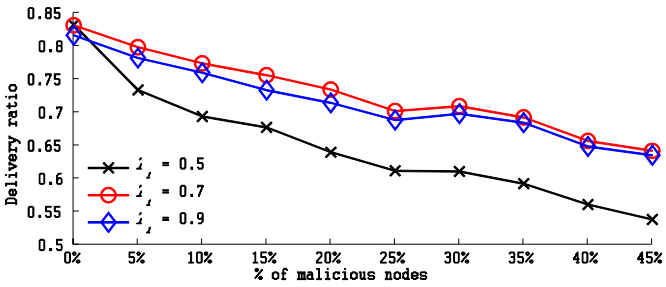


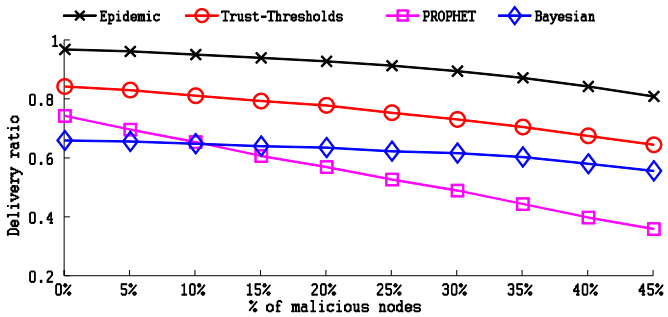
Fig. 4. Effect of T_f on delivery ratio.

6.3 Best Application-Level Trust Optimization Design Settings to Maximize Application Performance

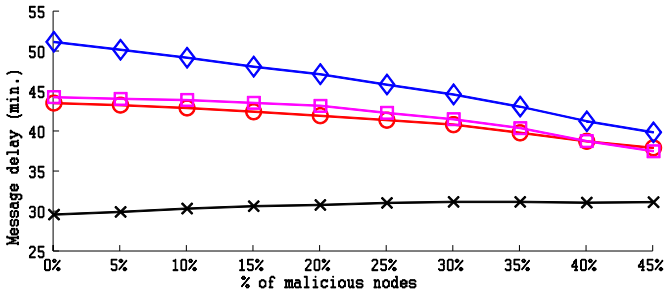
In this section, we apply the application-level trust optimization design in terms of the best minimum trust threshold T_{rec} for the selection of recommenders and the best message carrier trust threshold T_f to maximize delivery ratio in response to changing hostility reflected by the percent of malicious nodes. Fig. 4 shows delivery ratio versus T_f with the percentage of malicious nodes varying in [0-45 percent]. We set the trust recommender threshold, T_{rec}, at 0.6 to isolate out its effect. We notice that there is an optimal T_f value under which delivery ratio is maximized. With the environment setting (30 percent selfish nodes and 0 to 45 percent malicious nodes), the optimal value T_f value is around 0.7. The reason is that using a higher value of T_f helps generate a higher message delivery ratio by choosing only the most trustworthy nodes as message carriers, but it also introduces a higher message delay. Therefore, $T_f = 0.7$ is the best setting to balance the tradeoff between message delivery ratio versus message delay, except for the case when there are little malicious nodes for which $T_f = 0.5$ is the best setting.

6.4 Comparative Analysis

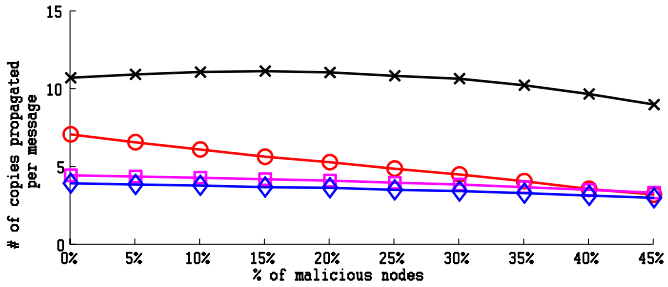
Lastly we conduct a comparative analysis, contrasting our trust-based protocol operating under the best settings identified with Bayesian trust-based routing [12], [14] and non-trust based (PROPHET [19] and epidemic [27]) protocols. PROPHET [19] uses the history of encounters and transitivity to calculate the probability that a node can deliver a message to a particular destination; it is considered as a benchmark "non-trust based" forwarding algorithm for DTNs in the literature. Bayesian trust-based routing on the other hand relies on the use of trust information maintained by a Bayesian based trust management system (such as a Beta reputation system [12], [14]) to make routing decisions. In a Bayesian trust management system, the trust value is assessed using the Bayes estimator, updated by both direct observations and indirect recommendations. The direct observations are directly used to update the number of positive and negative observations, whereas the recommendations are discounted by the confidence [12] or belief [14] of the trustor toward the recommender. Under Bayesian trust-based routing, a node is chosen as the message carrier only if its trust value is in the top Ω percentile and higher than the message carrier trust threshold T_f. We choose Bayesian



(a) Delivery Ratio.



(b) Message Delay.



(c) Message Overhead.

Fig. 5. Performance comparison (analytical results based on SWIM mobility).

trust-based routing because of its popularity in trust/reputation systems. We again consider double-copy forwarding (with $L = 2$) with nodes following the SWIM mobility model. For our trust-based secure routing protocol, we use the best settings for double-copy forwarding as identified earlier. There is no specific protocol parameter being used in epidemic routing. In epidemic routing, a message carrier forwards a message to every encountering node whenever this node has not seen the message before. It is selected as a baseline protocol to provide a performance bound in message delivery ratio and message delay. For PROPHET, the parameter values of *initialization constant*, *aging constant*, and *scaling constant* are 0.75, 0.25, and 0.98, respectively as suggested in [19], and we verify that PROPHET performs the best under these parameter settings through simulation. For the Bayesian trust model, there is no direct trust and indirect trust weight parameters because the weight to indirect trust is determined by confidence or belief [12], [14], [17] based on the positive and negative experiences/recommendations received. For fair comparison, we also use the best application-level protocol setting (i.e., T_f). When applying the Bayesian trust model to DTN routing.

Fig. 5 compares the message delivery ratio, delay, and overhead generated by our trust protocol against Bayesian trust-based, PROPHET, and epidemic routing protocols. The results demonstrate that our trust-based secure routing protocol designed to maximize delivery ratio can effectively trade off message overhead for a significant gain in delivery ratio. In particular, our protocol and Bayesian trust-based routing have less performance degradation in message delivery ratio than PROPHET when the percentage of malicious nodes increases. The reason is that using trust to select the next message carrier can avoid messages being forwarded to malicious nodes and then being dropped. Further, our trust-based routing protocol outperforms Bayesian trust-based routing and PROPHET in delivery ratio as it applies the best trust formation out of social and QoS trust properties. Moreover, our trust-based routing protocol also outperforms Bayesian trust-based and PROPHET in message delay except when there is a very high percent of malicious nodes (e.g., 40-45 percent of malicious nodes) in the system. The reason is that when there is a high percent of malicious nodes, our protocol tends to use a higher weight for healthiness and consequently a lower weight for connectivity, thus causing a higher message delay. Here we note that there is a tradeoff between message delivery ratio and message delay. When the percentage of malicious nodes in the network increases, a message originally successfully delivered with a longer message delay is more likely to be dropped; hence, this dropped message would not be counted in calculating message delay. This certainly does not mean we should have more malicious nodes in the network since the message delivery ratio will decrease. The similar observations appear in [19] investigating the performance of both message delivery ratio and message delay in DTN routing. Lastly, the message overhead of our trust-based routing protocol is significantly lower than epidemic routing. We conclude that our trust-based protocol approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message overhead.

7 Simulation Validation

In this section, we validate analytical results through extensive simulation using ns-3 [1]. The simulated DTN environment is setup as described in Table 1. We simulate two mobility patterns: a synthetic mobility model based on SWIM [15] and real mobility traces from [24], namely *Intel*, *Cambridge*, *Infocom05* and *Infocom06*. The simulation results obtained based on both SWIM mobility and mobility traces correlate well with analytical results in Fig. 5. We also present simulation results to demonstrate trust assessment accuracy, convergence and resiliency properties of our protocol. We refer the readers to Appendix B of the supplemental file [6] for detail.

8 Dynamic Trust Management

In this section, we perform a comparative analysis of our dynamic trust management protocol for DTN routing against PROPHET, Bayesian trust-based routing, and epidemic routing, all operating under best protocol settings dynamically in response to hostility changes over time. We consider two mobility patterns: the SWIM mobility model [15] and the *infocom06* mobility trace [24] to demonstrate the

effectiveness of our dynamic trust management protocol regardless of the mobility pattern. We refer the readers to Appendix C of the supplemental file [6] for detail.

9 Conclusion

In this paper, we designed and validated a trust management protocol for DTNs and applied it to secure routing to demonstrate its utility. Our trust management protocol combines QoS trust with social trust to obtain a composite trust metric. Our design allows the best trust setting (β, λ_d) for trust aggregation to be identified so that subjective trust is closest to objective trust for each individual trust property for minimizing trust bias. Further, our design also allows the best trust formation (w^X) and application-level trust settings (T_f, T_{rec}) to be identified to maximize application performance. We demonstrated how the results obtained at design time can facilitate dynamic trust management for DTN routing in response to dynamically changing conditions at runtime. We performed a comparative analysis of trust-based secure routing running on top of our trust management protocol with Bayesian trust-based routing and non-trust-based routing protocols (PROPHET and epidemic) in DTNs. Our results backed by simulation validation demonstrate that our trust-based secure routing protocol outperforms Bayesian trust-based routing and PROPHET. Further, it approaches the ideal performance of epidemic routing in delivery ratio and message delay without incurring high message or protocol maintenance overhead.

There are several future research areas including (a) exploring other trust-based DTN applications with which we could further demonstrate the utility of our dynamic trust management protocol design; (b) designing trust management for DTNs considering social communities and performing comparative analysis with more recent works such as [2], [3]; (c) implementing our proposed dynamic trust management protocol on top of a real DTN architecture [5] to further validate the protocol design, as well as to quantify the protocol overhead; (d) investigating trust-based admission control strategies as in [7], [8], [9] used by selfish nodes to maximize their own payoffs while contributing to DTN routing performance; and (e) developing trust and security management protocols for delay-tolerant, self-contained message forwarding applications based on the information-centric networks (ICN) architecture [13].

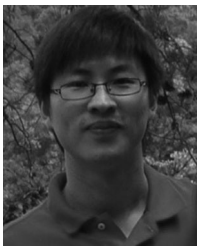
Acknowledgments

This work was supported in part by the Army Research Office under Grant W911NF-12-1-0445.

References

[1] "The ns-3 Network Simulator," http://www.nsnam.org/, Nov. 2011.
[2] E. Ayday, H. Lee, and F. Fekri, "Trust Management and Adversary Detection for Delay Tolerant Networks," *Proc. Military Comm. Conf.*, pp. 1788-1793, 2010.
[3] E. Ayday, H. Lee, and F. Fekri, "An Iterative Algorithm for Trust Management and Adversary Detection for Delay Tolerant Networks," *IEEE Trans. Mobile Computing*, vol. 11, no. 9, pp. 1514-1531, Sept. 2012.
[4] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine, "Maxprop: Routing for Vehicle-Based Disruption-Tolerant Networking," *Proc. IEEE INFOCOM*, pp. 1-11, Apr. 2006.
[5] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," *RFC 4838*, IETF, 2007.
[6] I.R. Chen, F. Bao, M. Chang, and J.H. Cho, "Supplemental Material for 'Dynamic Trust Management for Delay Tolerant Networks and Its Application to Secure Routing'," *IEEE Trans. Parallel and Distributed Systems*, 2013.
[7] I.R. Chen and T.H. Hsi, "Performance Analysis of Admission Control Algorithms Based on Reward Optimization for Real-Time Multimedia Servers," *Performance Evaluation*, vol. 33, no. 2, pp. 89-112, 1998.
[8] S.T. Cheng, C.M. Chen, and I.R. Chen, "Dynamic Quota-Based Admission Control with Sub-Rating in Multimedia Servers," *Multimedia Systems*, vol. 8, no. 2, pp. 83-91, 2000.
[9] S.T. Cheng, C.M. Chen, and I.R. Chen, "Performance Evaluation of an Admission Control Algorithm: Dynamic Threshold with Negotiation," *Performance Evaluation*, vol. 52, no. 1, pp. 1-13, 2003.
[10] J.H. Cho, A. Swami, and I.R. Chen, "A Survey on Trust Management for Mobile Ad Hoc Networks," *IEEE Comm. Surveys & Tutorials*, vol. 13, no. 4, pp. 562-583, Fourth Quarter 2011.
[11] E.M. Daly and M. Haahr, "Social Network Analysis for Information Flow in Disconnected Delay-Tolerant MANETs," *IEEE Trans. Mobile Computing*, vol. 8, no. 5, pp. 606-621, May 2009.
[12] M.K. Denko, T. Sun, and I. Woungang, "Trust Management in Ubiquitous Computing: A Bayesian Approach," *Computer Comm.*, vol. 34, no. 3, pp. 398-406, 2011.
[13] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking Named Content," *Comm. ACM*, vol. 55, no. 1, pp. 117-124, 2012.
[14] A. Jøsang and R. Ismail, "The Beta Reputation System," *Proc. Bled Electronic Commerce Conf.*, pp. 1-14, June 2002.
[15] S. Kosta, A. Mei, and J. Stefa, "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking," *Proc. Seventh IEEE Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks*, June 2010.
[16] F. Li, J. Wu, and A. Srinivasan, "Thwarting Blackhole Attacks in Disruption-Tolerant Networks Using Encounter Tickets," *Proc. IEEE INFOCOM*, pp. 2428-2436, 2009.
[17] N. Li and S.K. Das, "RADON: Reputation-Assisted Data Forwarding in Opportunistic Networks," *Proc. Second ACM Int'l Workshop Mobile Opportunistic Networking*, pp. 8-14, Nov. 2010.
[18] Q. Li, S. Zhu, and G. Cao, "Routing in Socially Selfish Delay Tolerant Networks," *Proc. IEEE INFOCOM*, pp. 1-9, Mar. 2010.
[19] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.*, vol. 7, no. 3, pp. 19-20, 2003.
[20] L. McNamara, C. Mascolo, and L. Capra, "Media Sharing Based on Colocation Prediction in Urban Transport," *Proc. 14th Ann. Int'l Conf. Mobile Computing and Networking*, 2008.
[21] A. Mei and J. Stefa, "Give2Get: Forwarding in Social Mobile Wireless Networks of Selfish Individuals," *Proc. IEEE Int'l Conf. Distributed Computing Systems*, pp. 488-297, June 2010.
[22] J.D. Musa, "Operational Profiles in Software-Reliability Engineering," *IEEE Software*, vol. 10, no. 2, pp. 14-32, Mar. 1993.
[23] I. Psaras, L. Wood, and R. Tafazolli, "Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges," technical report, Dept. of Electrical Eng., Univ. of Surrey, 2009.
[24] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD Data Set Cambridge/Haggle (v. 2009-05-29)," http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.
[25] S. Trifunovic, F. Legendre, and C. Anastasiades, "Social Trust in Opportunistic Networks," *Proc. IEEE INFOCOM*, pp. 1-6, Mar. 2010.
[26] K.S. Trivedi, "Stochastic Petri Nets Package User's Manual," Dept. of Electrical and Computer Eng. Duke Univ., 1999.
[27] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad Hoc Networks," technical report, Duke Univ., 2000.

Ing-Ray Chen received the BS degree from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, wireless systems, security, trust management, data management, real-time intelligent systems, and reliability and performance analysis. He currently serves as an editor for *IEEE Communications Letters, IEEE Transactions on Network and Service Management, Wireless Personal Communications*, *Wireless Communications and Mobile Computing*, *The Computer Journal*, *Security and Network Communications*, and *International Journal on Artificial Intelligence Tools*. He is a member of the IEEE and ACM.



Fenye Bao received the BS degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2006 and the ME degree in software engineering from Tsinghua University, Beijing, China in 2009. He received his PhD degree in Computer Science from Virginia Tech in 2013. Currently he is a technical staff member of LinkedIn. His research interests include trust management, security, wireless networks, wireless sensor networks, mobile computing, and dependable computing.

MoonJeong Chang received the BS, MS, and PhD degrees in computer science from Ewha Womans University in 2001, 2003, and 2009, respectively. Her research interests include mobility management, trust management, mobile computing, delay tolerant computing, and secure and dependable computing. She is currently a postdoc in the Department of Computer Science at Virginia Tech.

Jin-Hee Cho received the BA degree from the Ewha Womans University, Seoul, Korea and the MS and PhD degrees in computer science from the Virginia Tech. She is currently a computer scientist at the US Army Research Laboratory (USARL), Adelphi, Maryland. Her research interests include wireless mobile networks, mobile ad hoc networks, sensor networks, secure group communications, group key management, network security, intrusion detection, performance analysis, trust management, cognitive networks, social networks, dynamic networks, and resource allocation. She is a member of the IEEE and ACM.